Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Departement of Computer Science
Markus Püschel, David Steurer
Karel Kubicek, Johannes Lengler

25 November 2019

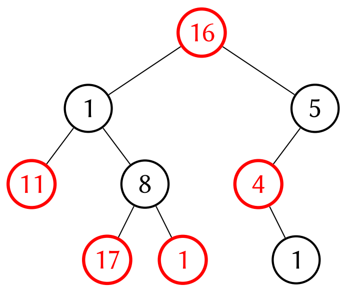
Algorithms & Data Structures Programming task 11 HS 19

The solutions for this sheet need to be submitted to the judge by **Monday, 9 December, 23:59:59**. Any minor changes to the template will be announced at Moodle: `https://moodle-app2.let.ethz.ch/mod/forum/discuss.php?d=41188`.

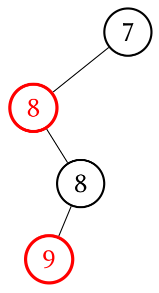
Exercise 11.1 *Tree cover* **(2 bonus point)**.

In this exercise, you are given a binary tree that stores integers (it is not a search tree, so the values stored in the tree and branches' height have no special property). Your task is to implement method `getMaximum` that returns the maximum sum of nodes (by nodes we mean their values) that satisfy the following property: *No two adjacent nodes can be included in the sum.*

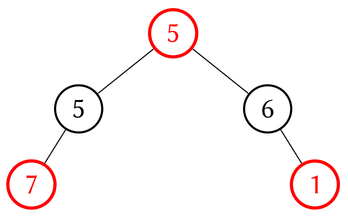
For illustration, there are some examples (red nodes are the selected nodes for the maximal sum):



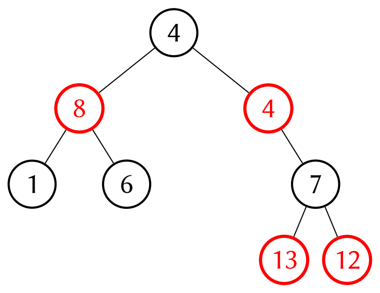
maximal sum is 49



maximal sum is 17



maximal sum is 13



maximal sum is 37

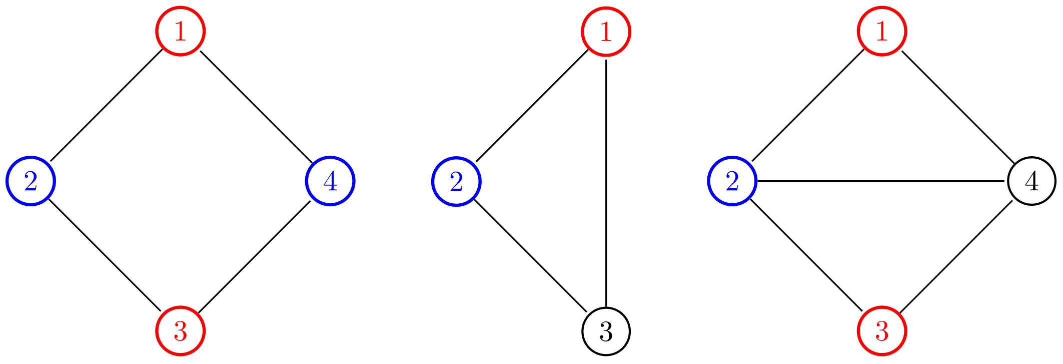
You get one point for each passing test set. The test set `small` does not test time complexity. That is done by the test set `large`. JUnitTest `testSingle` is meant for debugging, it can run a single instance that you struggle with, and visualize the respective tree.

Submission: Submit your `Main.java` at `https://judge.inf.ethz.ch/team/websubmit.php?cid=28784&problem=AD19H11P`. The enrollment password is "`asymptotic`".

Exercise 11.2 *Is two-colorable?* **(2 bonus point)**.

In this task, you will work with undirected graph given by an adjacency matrix. Your task is to test, if it is possible to label the graph using two colors, such that no two adjacent nodes have the same color.

For illustration, there are some examples:



The first graph is two-colorable, while the remaining two are not. In the second graph, after coloring node 1 and 2, we cannot use either red or blue for node 3, such that it would not be adjacent to a node of the same color. The same reasoning applies to node 4 in the last graph.

You get one point for each passing test set. The test set `small` does not test time complexity. That is done by the test set `large`. JUnitTest `testSingle` is meant for debugging, it can run a single instance that you struggle with, and visualize the respective graph.

Submission: Submit your `Main.java` at `https://judge.inf.ethz.ch/team/websubmit.php?cid=28784&problem=AD19H11Q`. The enrollment password is “`asymptotic`”.

As bonus exercises, we extend the tasks above.

Exercise 11.3 *Extension of Tree cover* **(additional exercise, no bonus points)**.

In the template TreeCoverExtended, you have to solve (almost) the same problem as in Exercise 11.1, but this time you should also find the set the nodes, not just their value. This time you should maximise the product of the involved nodes, not their sum. E.g., for the first example above, the maximising set is still the same, so your answer should be 11968 $(= 16 \cdot 11 \cdot 4 \cdot 17 \cdot 1)$.

You can check your solution via Judge by computing the product of your set and giving out this product and the array of nodes that contribute to this product. We will design the test cases in such a way that you only get a correct answer if your product is correct. (This is the reason to use products rather than sums.) The result must be stored as `BigInteger`.

There are two test sets. The test set `max` tests the maximal product, while the nodes that contribute to this product are tested by test set `nodes`. The same notation can be found in the local tests. JUnitTest `testSingle` is meant for debugging, it can run a single instance that you struggle with, and visualize the respective tree.

Submission: Submit your `Main.java` at `https://judge.inf.ethz.ch/team/websubmit.php?cid=28784&problem=AD19H11R`. The enrollment password is "`asymptotic`".

Exercise 11.4 *Additional graph properties* **(additional exercise, no bonus points)**.

For this task, we have no test set on Judge, neither by JUnitTest (the latter can be updated, check the course page news).

If you need more practice, you can solve the following tasks for what are known efficient algorithms. All graphs are undirected for the following problems.

Triangle count For a graph G, determine how many triangles G contains? A triangle is a set of three different vertices u, v, w such that all theedges (u, v), (v, w), (w, u) exist in G. Note that, e.g., $\{u, v, w\}$ and $\{u, w, v\}$ are just two different notations for the same set, so you must not count this triangle twice.

Count components For a graph G, how many conected components (Zusammenhangskomponenten) does the graph contain? As a reminder, a connected component is a maximal subgraph in which any two vertices are connected to each other by paths.

Is acyclic? For a given graph G, decide whether it is acyclic or not.

Graph diameter Let $G = (V, E)$ be a connected graph. For two vertices u and v, the graph distance $d(u, v)$ is the minimal number of edges that a path from u to v has. The graph diameter of G is $\max_{u,v \in V} d(u, v)$, i.e., the maximal distance between two vertices of the graph. Your task is to compute the diameter of G.